

Name: _____

Lab Section: _____

NET2006: Object-Oriented Programming

Lab 3: Working with Classes

Overview

This is an *in-class, individual* lab assignment. There are several goals. You will:

- Build a project that has multiple source files (three in total, two have the .cpp extension and one has the .h).
- Identify and correct syntax errors.
- Trace program execution to understand how objects are organized and how they behave.

I have placed copies of this lab's source code on my website.

Details: Create a Project

Here is the program. It is also available on my website (so that you don't have to type it in). The program consists of three separate source files: two .cpp files and one .h file.

File Containing Class Specification and main()

```
// Lab 3-circlesBuggy.cpp
// circles as graphics objects
#include "msoftcon.h" // for graphics functions
//*****
// class circle
//*****
class circle { // graphics circle
private:
// Data members hold current state of object (location, size, color, etc.)
int nCentreX, nCentreY; // Coordinates of center
int nRadius; // Radius of circle
color nFillColor; // Color
fstyle nFillStyle; // Fill pattern
public:
// Member functions implement the behaviour of objects
void Set(int nX, int nY, int nR, color nFC, fstyle nFS) // Sets circle attributes
{
nCentreX = nX;
nCentreY = nY;
nRadius = nR;
nFillColor = nFC;
nFillStyle = nFS;
}
void Draw() // draws the circle, using the object's data members
{
SetColor(nFillColor); // set color
SetFillStyle(nFillStyle); // set fill
DrawCircle(nCentreX, nCentreY, nRadius); // Draw solid circle
}
}
//*****
// void main()
//*****
void main()
{
InitializeGraphics(); // Initialize graphics system

circle c1; // Create three circle objects
circle c2;
circle c3;

// set circle attributes
c1.set(15, 7, 5, cBLUE, X_FILL);
c2.set(41, 12, 7, cRED, O_FILL);
c3.set(65, 18, 4, cGREEN, MEDIUM_FILL);

c1.draw(); // Draw circles
c2.draw();
c3.draw();
SetCursorPosition(1, 25); // Lower left corner on a standard console screen
}
```

The preceding file is based on an example program in your textbook. This version of the file has several syntax errors which will be flagged by the compiler when you finally try to build the project.

The following two files (one .cpp and one .h) are free of any errors. These two files give you access to some primitive console-oriented drawing routines.

File Containing Function Declarations for Graphics

```
//msoftcon.cpp -- provides routines to access Windows console functions
#include "msoftcon.h"

// Global Variables: BAD DESIGN!!!!!!!!!!!!
HANDLE hConsole; // Console handle
char cFillChar; // Character used for fill

//*****
// void InitializeGraphics()
//*****
void InitializeGraphics()
{
hConsole = CreateFileA(LPCSTR("CONOUT$"), GENERIC_WRITE | GENERIC_READ,
FILE_SHARE_READ | FILE_SHARE_WRITE, 0L, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL,
0L); //open i/o channel to console screen

COORD coordConsoleSize = {80, 25};
SetConsoleScreenBufferSize(hConsole, coordConsoleSize); //set to 80x25 screen size

SetConsoleTextAttribute(hConsole, (WORD)((0 << 4) | 15)); //set text to white on black

cFillChar = 'xDB'; //default fill is solid block
ClearScreen();
} // end InitializeGraphics()

//*****
// void SetColor(color cForeground, color cBackground)
//*****
void SetColor(color cForeground, color cBackground)
{
SetConsoleTextAttribute(hConsole, (WORD)((cBackground << 4) | cForeground));
} //end SetColor()

/* 0 Black 8 Dark gray
1 Dark blue 9 Blue
2 Dark green 10 Green
3 Dark cyan 11 Cyan
4 Dark red 12 Red
5 Dark magenta 13 Magenta
6 Brown 14 Yellow
7 Light gray 15 White
*/

//*****
// void SetCursorPosition(int nX, int nY)
//*****
void SetCursorPosition(int nX, int nY)
{
COORD coordCursorPosition; //origin in upper nLeft corner
coordCursorPosition.X = nX - 1; //Windows starts at (0, 0)
coordCursorPosition.Y = nY - 1; //we start at (1, 1)
SetConsoleCursorPosition(hConsole, coordCursorPosition);
} // end SetCursorPosition()

//*****
// void ClearScreen()
//*****
void ClearScreen()
{
SetCursorPosition(1, 25);
for(int j=0; j<25; j++)
_putchar("\n");
SetCursorPosition(1, 1);
} //end ClearScreen()
```

```

//*****
// void Wait(int milliseconds)
//*****
void Wait(int milliseconds)
{
Sleep(milliseconds);
} // end Wait()

//*****
// void ClearLine() //clear to end of line
//*****
void ClearLine() //clear to end of line
{
// //80 spaces
// 0.....1.....2.....3.....4
// 12345678901234567890123456789012345678901234567890
_cputs(" ");
_cputs(" ");
} // end ClearLine()

//*****
// void DrawRectangle(int nLeft, int nTop, int nRight, int nBottom)
//*****
void DrawRectangle(int nLeft, int nTop, int nRight, int nBottom)
{
char szTemp[80];
int nWidth = nRight - nLeft + 1;

int j;
for(j=0; j<nWidth; j++) //string of squares
szTemp[j] = cFillChar;
szTemp[j] = '\0'; //null character

for(int nY=nTop; nY<=nBottom; nY++) { //stack of strings
SetCursorPosition(nLeft, nY);
_cputs(szTemp);
}
} // end DrawRectangle()

//*****
// void DrawCircle(int nCenterX, int nCenterY, int nRadius)
//*****
void DrawCircle(int nCenterX, int nCenterY, int nRadius)
{
double dTheta, dIncrement, xF, pi=3.14159;
int nX, xN, yN;

dIncrement = 0.8 / static_cast<double>(nRadius);
for(dTheta=0.0; dTheta<=pi/2.0; dTheta+=dIncrement) { //quarter circle
xF = nRadius * cos(dTheta);
xN = static_cast<int>(xF * 2 / 1); //pixels not square
yN = static_cast<int>(nRadius * sin(dTheta) + 0.5);
nX = nCenterX-xN;
while(nX <= nCenterX+xN) { //fill two horizontal lines, one for each half circle
SetCursorPosition(nX, nCenterY-yN); _putch(cFillChar); //nTop
SetCursorPosition(nX++, nCenterY+yN); _putch(cFillChar); //nBottom
} // end while
} //end for
} // end DrawCircle()

//*****
// void DrawLine(int nX1, int nY1, int nX2, int nY2)
//*****
void DrawLine(int nX1, int nY1, int nX2, int nY2)
{
int w, z, t, w1, w2, z1, z2;
double dDeltaX=nX1-nX2, dDeltaY=nY1-nY2, dSlope;
bool bMoreHorizontal;

if( fabs(dDeltaX) > fabs(dDeltaY) ) { //more horizontal
bMoreHorizontal = true;
dSlope = dDeltaY / dDeltaX;
w1=nX1; z1=nY1; w2=nX2, z2=nY2; //w=nX, z=nY
}
else { //more vertical
bMoreHorizontal = false;
dSlope = dDeltaX / dDeltaY;
w1=nY1; z1=nX1; w2=nY2, z2=nX2; //w=nY, z=nX
}

if(w1 > w2) { //if backwards w
t=w1; w1=w2; w2=t; // swap (w1,z1)
t=z1; z1=z2; z2=t; // with (w2,z2)
}
for(w=w1; w<=w2; w++) {
z = static_cast<int>(z1 + dSlope * (w-w1));
if( !(w==80 && z==25) ) { //avoid scroll at 80,25

```

```

if(bMoreHorizontal)
SetCursorPosition(w, z);
else
SetCursorPosition(z, w);
_putch(cFillChar);
}
} // end DrawLine()

//*****
// void DrawPyramid(int nX1, int nY1, int nHeight)
//*****
void DrawPyramid(int nX1, int nY1, int nHeight)
{
int nX, nY;
for(nY=nY1; nY<=nY1+nHeight; nY++) {
int nIncrement = nY - nY1;
for(nX=nX1-nIncrement; nX<=nX1+nIncrement; nX++) {
SetCursorPosition(nX, nY);
_putch(cFillChar);
}
}
} // end DrawPyramid()

//*****
// void SetFillStyle(fstyle fsNewFillStyle)
//*****
void SetFillStyle(fstyle fsNewFillStyle)
{
switch(fsNewFillStyle) {
case SOLID_FILL: cFillChar = 'xDB'; break;
case DARK_FILL: cFillChar = 'xB0'; break;
case MEDIUM_FILL: cFillChar = 'xB1'; break;
case LIGHT_FILL: cFillChar = 'xB2'; break;
case X_FILL: cFillChar = 'X'; break;
case O_FILL: cFillChar = 'O'; break;
}
} // end SetFillStyle()

```

File Containing Function Declarations for Graphics

```

//msoftcon.h
//declarations for Lafore's console graphics functions
//uses Window's console functions

#ifndef _INC_WCONSOLE // Don't let this file be included
#define _INC_WCONSOLE // Twice in the same source file

#include <windows.h> // For Windows console functions
#include <conio.h> // For kbhit(), getch()
#include <math.h> // For sin, cos

enum fstyle { SOLID_FILL, X_FILL, O_FILL, LIGHT_FILL, MEDIUM_FILL, DARK_FILL };

enum color {
cBLACK=0, cDARK_BLUE=1, cDARK_GREEN=2, cDARK_CYAN=3,
cDARK_RED=4, cDARK_MAGENTA=5, cBROWN=6, cLIGHT_GRAY=7,
cDARK_GRAY=8, cBLUE=9, cGREEN=10, cCYAN=11,
cRED=12, cMAGENTA=13, cYELLOW=14, cWHITE=15};

//*****
// Function Declarations (Function Prototypes)
void InitializeGraphics();
void SetColor(color cFG, color cBG = cBLACK);
void SetCursorPosition(int nX, int nY);
void ClearScreen();
void Wait(int nMilliseconds);
void ClearLine();
void DrawRectangle(int nLeft, int nTop, int nRight, int nBottom);
void DrawCircle(int nX, int nY, int nRadius);
void DrawLine(int nX1, int nY1, int nX2, int nY2);
void DrawPyramid(int nX1, int nY1, int nHeight);
void SetFillStyle(fstyle fsNewFillStyle);
#endif /* _INC_WCONSOLE */

```

Adding Files to the Project

You must add all .cpp files to the project (in this case **Lab3-3-circlesBuggy.cpp** and **msoftcon.cpp**). The header file **msoftcon.h** will be managed automatically because of the **#include** directive in the source code. Nevertheless, you can add it to the *Header Files* section of the project.

When you try compiling, you will encounter syntax errors (intentional errors to give you practice in finding them).

Details: Dealing with the Syntax Errors

As is commonly the case, the compiler doesn't recognize the error until long after the error occurred. So the correction must take place much earlier.

(3 marks) When you successfully compile the corrected program, identify the syntax corrections directly in the program code in this document.

Details: Inspecting Objects at Runtime

To understand how object-oriented programming works, you need to understand how data is organized in memory, and how the object-oriented functions access that data. Let's start that process of learning.

Set a breakpoint on the statement that declares and defines the first **circle** object. When program execution reaches the breakpoint, you'll need to open two debug windows: *Watch* and *Autos*. You have used the *Watch* window before; it allows you to enter variable names explicitly. The *Autos* window (as the name suggests) automatically shows you variables that are relevant at the current execution point. This window is important for this lab because it tells you something about how object-oriented functions work.

(1 mark) What is the address of **c1**?

(1 mark) What is the address of **c2**?

(1 mark) What is the address of **c3**?

Step into the first call to the object-oriented function at the statement `c1.Draw();`. To see the variables in his function in the *Autos* window, you need to step onto the first line of executable code (that is, one line after the opening brace `{`).

(1 mark) What does the *Autos* window show (list the names and the contents of variables shown)?

(1 mark) In what way does this correlate with the information you found in the first set of three questions in this lab?

Step into the second call to the object-oriented function at the statement `c2.Draw();`.

(1 mark) What does the *Autos* window show (list the names and the contents of variables shown)?

(1 mark) In what way does this correlate with the information you found in the first set of three questions in this lab?

Step into the third call to the object-oriented function at the statement `c3.Draw();`.

(1 mark) What does the *Autos* window show (list the names and the contents of variables shown)?

(1 mark) In what way does this correlate with the information you found in the first set of three questions in this lab?

Here's the *publicly accessible source code* for `DrawRectangle()`.

```
void DrawRectangle(int nLeft, int nTop, int nRight, int nBottom);
```

Take a close look at the specification for the `circle` class. Your newly created `rectangle` class will be very similar. But there are a few minor differences:

- A `circle` object needed to track its centre (with x,y coordinates), and its radius: a total of 3 `int` values.
- A `rectangle` object needs to track its top position (a y value), its left position (an x value), its bottom position (another y value), and its right position (another x value): a total of 4 `int` values.
- Your implementation of the `Set()` function for the `rectangle` class needs to reflect the change in variables described above.
- Your implementation of the `Draw()` function for the `rectangle` class needs to reflect the change in variables described above and call the primitive function `DrawRectangle()`.
- Finally, you need to create one or more `rectangle` objects in `main()`, call `Set()` for those newly created `rectangle` objects, and call `Draw()` for those newly created `rectangle` objects.

(10 marks) After successfully completing the implementation of the `rectangle` class, you are ready to submit this lab.

Be sure to print the `6-3-circles.cpp` file to document the implementation of the `rectangle` class.

Submission Process

- Print the corrected file with your circle and rectangle code. This revised source code will contain the implementation of the `rectangle` class and the calls to create `rectangle` objects, as well as `Set()` and `Draw()` these objects.
- Staple this source code to this lab document.
- Submit this by the beginning of the next lab period.
- Some of you may be able to finish this in the current lab period. (Entitling you to a bonus mark.)

Details: Create and Draw Other Objects

Open the file `msoftcon.h`. This a header file; header files are often thought of as the "*publicly accessible source code*". The detailed code that does all the real work is generally clustered in the associated `.cpp` file. This may have already been compiled and not be visible¹.

There are several functions available in this set of drawing routines. Try creating code to specify a `rectangle` class, which will, in turn call the primitive function `DrawRectangle()`.

¹ This was the case with the use of `rand()` and `getch()`. These two functions have already been implemented in a compiled, machine-code form. You have already used them many times. You don't ever see the original source code. But you can see the *publicly accessible source code* in the associated header file: `stdlib.h` for `rand()` and `conio.h` for `getch()`.